Contents

1. Introduction: How To Use This Handbook

Note: This document contains all of the technical requirements necessary to obtain the Designed for Windows NT® and Windows® 95 Logo. There are additional forms (Vendor Questionnaire, License Agreement, and Testing Agreement) you will need to complete in order to submit your application for testing. These are available at http://www.microsoft.com/windows/thirdparty/winlogo/.

This handbook is divided into the following sections:

What Does The "Designed For Windows NT and Windows 95" Logo Mean?
This section provides an overview of the purpose of the Logo, why this Logo has value to your customers and how it differs from the Designed for Windows 95 Logo.

How To Obtain The Logo
This section describes the procedure you must follow in order to apply for and obtain the Logo.

The Logo Requirements: Detailed Specifications
This section details the complete requirements that your application must meet in order to obtain the Logo.

Legend For Detailed Specifications

Type	Definition
Required	The feature must be supported as described in order for the application to pass testing and receive the Logo.
Exception	These are situations where a requirement may not apply.
Recommendation	These features are recommended but not required at this time in order for the application to pass testing and receive the Logo. A recommendation will become a requirement in the future.
Verification	These are suggested ways that you can test whether your application meets a specific requirement or recommendation.
Tip	These are helpful suggestions. They are not recommendations or requirements.
Note	These are clarifications to the requirements/recommendations.

Exceptions, Exemptions, And Additional Requirements
This section describes exceptions, exemptions, and additional requirements for certain application categories, such as non-file-based applications, development tools, utilities, games and multimedia applications, and add-on products.

Frequently Asked Questions (FAQ)
This section provides answers to the questions we find applicants ask most often about the Logo program.

Appendixes
The appendixes provide additional information to help you with your Windows-based application development, but they do not contain any requirements.

Key Definitions

Term	Definition
AutoPlay	Denotes the behavior of automatically launching a program immediately after inserting a CD-ROM into the drive.
Autorun.inf	Text-based information file called during AutoPlay. Contains operating system instructions describing what executable to launch, etc.
Core Components	Core components are defined as DLLs that application developers have come to depend on for the proper functionality of their applications. Reverting to an older version or to an incorrectly implemented version may break many third party applications. In general terms, the core components are USER, GDI, and KERNEL. Core components are files which are installed by the operating system during a full installation, or which are considered operating system upgrades (for example, DirectX™, Direct3D™). This includes such things as DLLs which are installed by applets. This list is dynamic. For the most up to date list of core components for Windows NT and Windows 95, see www.microsoft.com/windows/thirdparty/winlogo.
Degrade Gracefully	Does not crash the operating system (GPF or bluescreen) Dialog box or other visual and audio cue appears informing the user that the functionality is not available on *X* version of *Y* operating system User is not required to close the application, but may continue to use the other functionality

Term	Definition
File-Based Application	File-based applications have as their primary purpose the creation and editing of documents and include **Open**, **Save**, and **Close** commands, typically on the **File** menu of the application.
Labeling Scheme	Common to games, accounting, and database programs. Users save "profiles" of reports, game states, etc. which have limited naming schemes. Labels may create actual file names, but the file naming behavior only vaguely represents the input of the user. Example: User may enter, "File Name Test", but the application would save "SAV001.EXT", or "FILAAAA.EXT" to the hard disk. These files are not intended to be accessed directly from Windows Explorer. A labeling scheme is not required to save long file names to the hard disk and is exempt from the long file name requirements.
Non-File-Based Application	A non-file-based application is one that is *not* primarily used to create, edit, and save files (although file operations may be common ancillary tasks).
Product Category	These categories are used to determine what behavior is required from the application to pass the Logo test. Includes: File-Based, Non-File-Based, Utility, Development Tool, Add-On. The final determination of what category applies to a product will be made by Microsoft.
Refcount	The act of incrementing and decrementing shareable components in the system registry under the ...\SharedDLLs key.
Shared Components	Any files which are installed by an application, but might be shared by multiple applications. Shared components are usually DLLs, but may be EXEs or any other file type. These must be "refcounted" unless they are "core components" or installed to the application's own directory.
Suite vs. Product	A suite is a collection of programs typically denoted by more than one shortcut on the Start Menu. A product can be a suite or a single executable. The final determination of whether or not a product is a suite will be made by Microsoft and VeriTest.

Term	Definition
Telephony Centric	One of two categories of communications applications. An application that performs any telephony function beyond simple dialing of voice calls is considered a telephony *centric* application (including, for example, all applications that use a modem for data or fax calls). Examples of telephony centric include: dialers, answering machines, voicemail, interactive voice response, modem or fax communications, etc.
Telephony Enabled	One of two categories of communications applications. Telephone communications are not the primary purpose of the application. Examples of telephony enabled applications are: Personal Information Managers, address books, and databases
Win32	Any 32-bit executable file described as "[PE_Win32]" by the Exedump program available on the Win32® SDK (Software Development Kit).

2. What Does the "Designed for Windows NT and Windows 95" Logo Mean?

The Designed for Windows NT and Windows 95 Logo Program was developed by Microsoft to help end users easily identify desktop hardware and software products that were designed specifically for the Microsoft® Windows family of 32-bit operating systems, Windows NT Workstation (hereafter referred to as Windows NT) and Windows 95. Users can mix and match hardware and software products with the Designed for Windows NT and Windows 95 Logo and be assured that the products will take advantage of the new technologies integrated into both of these operating systems. As a software vendor, licensing the Logo enables you to use the Logo on your product packaging, advertising, collateral, and other marketing materials. This signals to customers that:

- your product is fully functional on both Windows NT and Windows 95; and
- you designed your product to give them extra benefits when it is run on either Microsoft Windows NT or Windows 95.

Operating System Versions To Which This Logo Applies

In order to receive the Designed for Windows NT and Windows 95 Logo, all requirements will be fully tested in parallel and must pass on the latest releases of both Windows NT Workstation (version 4.0 at the time of publication) and Windows 95. If your application runs *only* on Windows 3.X, *only* on Windows 95, or *only* on Windows NT, do not submit your application for this Logo. If your application runs on Windows NT 3.51, but not on Windows NT 4.0, do not submit your application for this Logo.

There is no Microsoft Logo program for Windows NT alone for software.

If your application runs only on Windows 95, you may choose to submit your application for the Designed for Windows 95 Logo Program instead. See the separate requirements for that program on http://www.microsoft.com/windows/thirdparty/winlogo//.

The Designed for Windows NT and Windows 95 Logo is a superset and evolution of the Designed for Windows 95 Logo Program. Although Microsoft will continue to offer the Designed for Windows 95 Logo at least through January 1, 1997, the Designed for Windows 95 Logo will eventually be phased out, and Microsoft will offer only the Designed for Windows NT and Windows 95 Logo.

Because the Designed for Windows NT and Windows 95 Logo is a superset of the Designed for Windows 95 Logo, passing the test for the Designed for Windows NT and Windows 95 Logo before January 1, 1997, automatically grants the vendor license to use the Designed for Windows 95 Logo as well.

Other Microsoft Logo Programs

The Designed for Windows NT and Windows 95 Logo is for commercially marketed desktop applications which run primarily on Windows NT Workstation and Windows 95. It is not for client/server applications or Windows NT Server-based applications. If your application is a client/server application or runs primarily on Windows NT Server, you should consider applying for the Designed for BackOffice™ Logo. Receiving the Designed for Windows NT and Windows 95 Logo *is not* a requirement for receiving the Designed for BackOffice Logo. For additional information on the BackOffice Logo, see http://www.microsoft.com/backoffice/designed, or send an e-mail message to bckoffc@microsoft.com.

Microsoft also offers the Microsoft Office Compatible Logo Program. The Designed for Windows NT and Windows 95 Logo *is* now a requirement for receiving the Microsoft Office Compatible Logo. For additional information on the Microsoft Office Compatible Logo program, see http://www.microsoft.com/msoffice/ofccomp, or send an e-mail to offcomp@microsoft.com.

Testing

Software applications are tested by an independent laboratory, VeriTest, Inc. (http://www.veritest.com or send an e-mail to logolab@veritest.com).

The Designed for Windows NT and Windows 95 Logo indicates that a software product provides all the features outlined in this handbook; it is not a "quality assurance" seal. Neither Microsoft nor the testing organization will be testing the quality of your product or ensuring that it is "bug-free," the testing organization's job is just to make sure that your product has full functionality, that the Logo features are present, and that your product is not generating frequent faults or system crashes.

Please note that the license agreement states:

You may only use the Logo as a symbol that your Product is compatible with Microsoft Windows NT and Windows 95. You may not explicitly state or imply that Microsoft or the testing organization in any way endorses your product. Also, the Windows NT and Windows 95 Logo program is not intended to be a "certification" program, that is, the Logo does not represent that Microsoft or the testing organization certifies your product(s) in any way.

3. How to Obtain the Logo

3.1. Process Overview Steps

1. Obtain a Pretest Kit.
   - The Pretest Kit is available on the Internet at: http://www.microsoft.com/windows/thirdparty/winlogo/.htm
   - For a physical copy, send an e-mail message requesting a Pretest Kit to winlogo@microsoft.com.
   - VeriTest and Microsoft license agreements may only be acquired within the physical Pretest Kit or by fax service:
     - The Microsoft Windows NT and Windows 95 Logo license agreement is available by phone/fax service at (206) 635-2222, document #830
     - VeriTest's testing agreement is available by phone/fax service at (206) 635-2222, document #831
2. Submit your product to VeriTest using the following Pretest materials. Your submission packet must include the following:
   - Your software product (on floppy disk, or CD-ROM)
   - Step-by-step instructions for using your product (submit entirely on paper)
   - The completed Vendor Questionnaire found in the Pretest Kit or online. The Vendor Questionnaire is a self-extracting application; after the file has been extracted, you must run the .exe file. The completed Vendor Questionnaire file must be submitted on disk.
   - The VeriTest Testing Agreement (submit on paper)
   - The Windows NT and Windows 95 Logo License Agreement (submit on paper)

     License Agreement. Your signature on these contracts does not mean that you can start using the Designed for Windows NT and Windows 95 Logo. It means that you've agreed to the license terms and are awaiting your product's passage of the tests and Microsoft's signing of the license.

     Confidentiality Option. On the Testing Agreement with VeriTest, you are given the option to keep your testing and test results confidential from Microsoft. If you choose this option, upon passing you must sign a release allowing VeriTest to send the results to Microsoft. Only after you sign this release can Microsoft grant you license to use the Logo

3. You will receive test results from VeriTest within eight (8) business days from when your complete application package is received. VeriTest will send both you and Microsoft a copy of the test results.
4. If your product passes testing, VeriTest will then send to Microsoft the license agreement that you previously signed and submitted with your product testing to VeriTest.
5. Please read the trademark guidelines (available in Pretest Kit) before using the Logo
6. Please see http://www.veritest.com for current fee schedule. All testing fees are paid to VeriTest, *not* Microsoft.
7. If your product box is printed before you obtain the license to use the Designed for Windows NT and Windows 95 Logo, you may order Logo sticker from Special Effects, an outside vendor. The stickers will be delivered when you receive the license to us the Logo.
8. Microsoft will sign the license agreement and send it to the person designated as the marketing contact in the Vendor Questionnaire you submitted. At this point, Microsoft will also send an artwork kit, which contains electronic and paper copies of the Logo and Logo usage guidelines.
9. If your company already has product(s) licensed to use the Designed for Windows NT and Windows 95 Logo, Microsoft will send you an update to the license after we receive your positive test results. This update will reflect the addition of the new product(s) to the license. Your signature on this update is required in order to complete the process.
10. The license allows you to use the Logo on packaging, advertising, and promotional materials.

3.2. Important Notes on International and Localized Versions

Because of the differences in core components of Windows NT and Windows 95 when localized to a language other than English, the Logo can only be used for localized versions of a product which are in the same language group (see below) as the one which was tested.

Double Byte Language Group - Far East: Japanese, Korean, Traditional Chinese (Taiwan), Simplified Chinese (PRC).

Single Byte Language Group - Western European: English, German, French, Spanish, Swedish, Italian, Dutch, Portuguese/Brazilian, Portuguese/Iberian, Catalan, Italian.

Single Byte Language Group - Eastern European: Finnish, Russian, Czech, Slovenian, Greek, Hungarian, Polish, Turkish, Slovak.

Single Byte Language Group - Other: Arabic, Hebrew, Thai, Vietnamese.

An application must be submitted in the primary language version in which it is to be marketed (where the most units are expected to be sold). VeriTest will test the application using the matching localized versions of Windows NT and Windows 95. If the application passes testing, the Logo will be licensed for use with that language version of the application only and other languages within the same language group only. For example, if an application is submitted in English and passes in English, the Logo may be used on the packaging and advertising for the English version and also any Western European language listed above. The Logo may not be used on the Japanese version for example, (unless the application is also submitted and passes in Japanese or another Double Byte language.)

If the Logo is desired for a version of the application which is in another language group, then a version of the application localized to a language in the intended group must be submitted and a retest fee will be applied to this testing. If the application passes testing in this new version, the Logo is licensed for use with that language version as well as other languages within the same language group.

You do not need to notify Microsoft if distributing the product internationally, unless distribution includes the PRC, Taiwan, or Korea. If you are distributing to these countries, please send the Windows Logo Department (winlogo@microsoft.com) a note with your company name, title(s) being distributed, and countries where they are being distributed. Please note the following language from the License Agreement:

The license right set forth in Section 2(a) shall not extend to the Republic of China ("Taiwan"), South Korea ("Korea"), or the People's Republic of China ("PRC"), unless and until COMPANY provides MS with written notice of COMPANY's intent to distribute Product in these countries. COMPANY agrees not to use the Logo in such countries and shall not be licensed pursuant to this Logo Agreement to do so until COMPANY has provided MS with such written notice.

4. The Logo Requirements: Detailed Specification

To qualify for the Designed for Windows NT and Windows 95 Logo, applications in all product categories must meet all requirements listed in this section (4.1 through 4.11), except where explicitly noted. All requirements apply and will be tested on the most recent releases of both Windows 95 and Windows NT, except where explicitly noted.

4.1. 32-bitness

Required:

☑ An application must be a Win32 application programming interface executable file compiled with a 32-bit compiler that generates an executable file of the PE (Portable Executable) format ("[PE_Win32]" as reported by the Exedump program on the Win32 SDK). This means that all the major program files (DLLs and EXEs) must be 32-bit with the exceptions stated below.

Note:. On Windows NT, thunks allow applications to call from 16-bit code to 32-bit code. On Windows 95, thunks allow applications to call in both directions. However, calling from 32-bit to 16-bit on Windows NT is not supported; you need 32-bit equivalent code for Windows NT. Fully 32-bit code allows applications to ship a single binary.

Exception:
If your application is not represented in PE format (interpreted code, for example), then the "run-time engine" must be a Win32-based executable file in the PE format. For example, if you develop an application in Microsoft Access, your application is an .mdb file, not an .exe, but Access.exe would need to be a Win32-based executable file in the PE format.

Exception:
Under the following circumstances, certain 16-bit code elements may be included in your product:

- Some minor, subsidiary 16-bit executable files and dynamic-link libraries (DLLs) may be used *in order to provide backward compatibility and links to 16-bit products.* However, your main program files (including DLLs) must be 32-bit.
- Other minor, subsidiary 16-bit executable files and DLLs may be used for *convenience* in a very limited fashion, however, *only* if there is no existing 32-bit version of the DLL you need for your application. These files cannot constitute a significant portion of the product. For example, imagine you wish to use 16-bit QuickTime® for playing

videos. *Can you still qualify for the Logo?* No. Since 32-bit QuickTime is now available, you must use the 32-bit version of QuickTime to qualify for the Logo. *What if you are shipping a commercially available 16-bit installer with your application?* If a 32bit version of that installer is not available, the inclusion of a 16-bit installer will not jeopardize Logo eligibility at this time. Note that this exception will likely be removed in future versions of the Logo requirements.

- You must fully explain any and all instances of 16-bit code when you submit your product for testing. This must be done in the electronic Vendor Questionnaire you submit with your application.

4.2. Distribution

Required:
In order for a bundled package of applications (commonly known as a "suite") to use the Logo, each individual product in that suite must be tested and pass the Logo requirements.

Required:
The Logo program is for 32-bit Windows-based applications. However, vendors may distribute 16-bit components on their distribution media (for example, the 16-bit version of your application so that the user can set up and run the application on Windows 3.*x* from the same CD-ROM) as long as the following conditions are met:

- The installer for the 16-bit elements must clearly distinguish the difference between these products and the 32-bit Designed for Windows NT and Windows 95 products on the same disks. A message such as "Do you also wish to install Product Y? This is a 16-bit product that is not designed for Microsoft Windows NT and Windows 95 but must operate in this environment" must be included.

Required:
Your installer must automatically detect the version of Microsoft Windows NT and/or Windows 95 running and install the correct version of your product automatically. This must be a seamless experience for the user.

Required:
Windows NT runs on multiple hardware platforms. Windows 95 is a single-platform operating system. You may wish to ship multiple platform versions (different binaries that work on platforms such as MIPS® R4000®, Intel® x86, Motorola® PowerPC™, or Alpha AXP™) of your application in the same package. All multiple platform versions may be submitted simultaneously for testing.

- If your product can be installed on multiple platforms from a single distribution media (for example, a CD-ROM), the installer must automatically detect the platform. It is not acceptable to ask the user which platform to install.

- You must include a version of your application that runs on Windows 95 in the multiple-platform package or in the advertising or marketing materials on which you are using the Logo.

4.3. Installation

Required:
☑ The product must have an installation program with a graphical Setup program that executes smoothly in a 32-bit Microsoft Windows operating system.

Required:
☑ The installer must lead the user through the process, not just instruct the user to copy or decompress files.

Required:
☑ The installer must make any configuration changes automatically, such as registry changes, etc.

Required:
☑ Products distributed on CD-ROM must utilize the AutoPlay feature in 32-bit Microsoft Windows operating system platforms to begin setup or launch the program itself the first time the application is run. It is up to the vendor whether AutoPlay is enabled on subsequent insertions of the CD-ROM.

Required:
☑ The installer for media other than CD-ROM must provide the ability to launch the installer through **Add/Remove Programs** in the Control Panel.

Required:
☑ The installer must make the following entries to the system Registry: Your company name, application name, and version number.

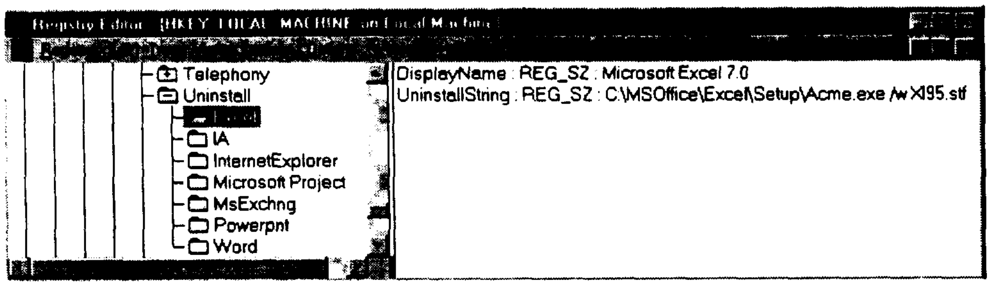
- You must adhere to the following convention:
  `[HKEY_LOCAL_MACHINE]\SOFTWARE\Your Company Name\Your Product Name`
- It is okay to create subcategories in the Registry, below the appropriate location, if this is necessary. For example, a company might want to group its game products under \company\games, etc.
- The uninstaller must be properly registered and must appear under **Add/Remove Programs** in the Control Panel. The method for this registration is:

```
[HKEY_LOCAL_MACHINE]\SOFTWARE\Microsoft\Windows\Current
Version\Uninstall\YourProductName

DisplayName=REG_SZ:<your product name and version number>
```

```
UninstallString=REG_SZ: c:\apps\myapp\uninstll.log /h
```

The following illustration is an example of the Uninstall String Executable Name located in the Windows NT Registry:

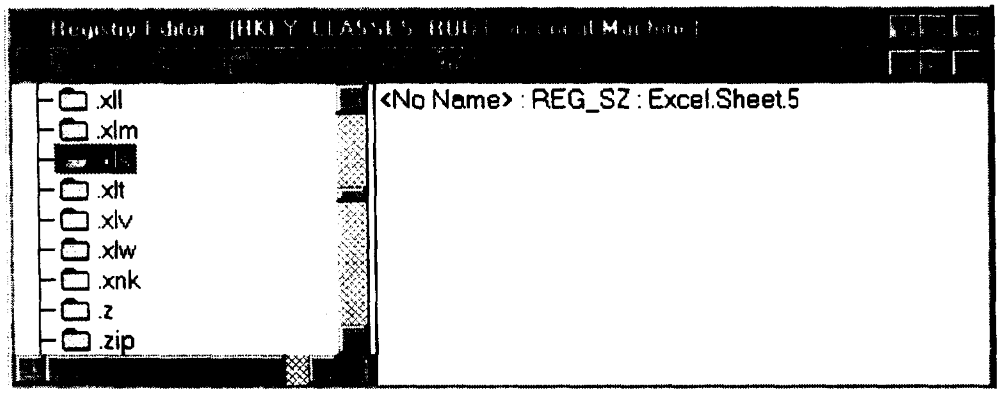


DisplayName : REG_SZ : Microsoft Excel 7.0
UninstallString : REG_SZ : C:\MSOffice\Excel\Setup\Acme.exe /w Xl95.stf

Native data file types (if applicable) must be registered as follows:

```
[HKEY_CLASSES_ROOT]
.(file type extension)
    (Default) = REG_SZ:FileTypeID
```

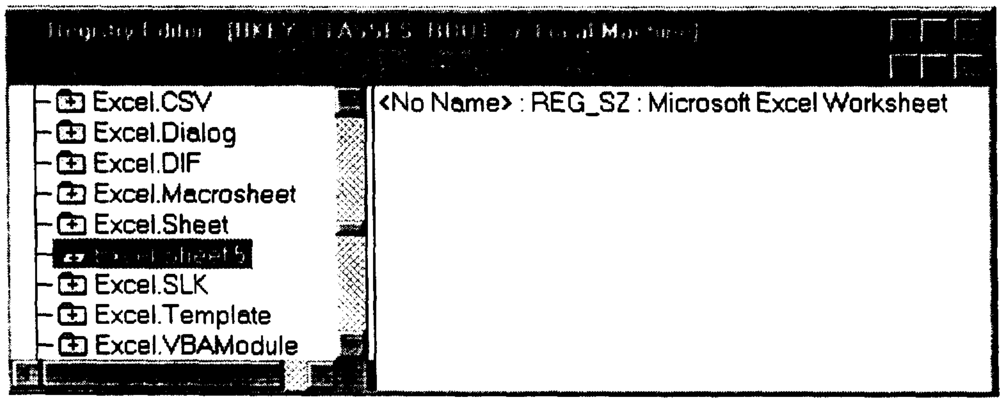
The following illustration is an example of the File Type Extension located in the Windows NT Registry:



<No Name> : REG_SZ : Excel.Sheet.5

```
FileTypeID=
      (Default) = REG_SZ:"Friendly Name"
```

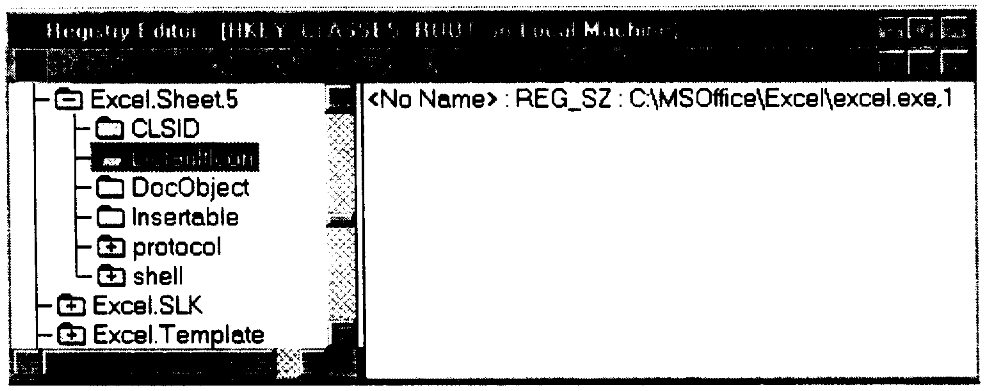
The following illustration is an example of the "Friendly Name" in the Windows NT Registry:



<No Name> : REG_SZ : Microsoft Excel Worksheet

```
DefaultIcon
(Default) = REG_SZ:"Path\FileName[,Index]"
```

The following illustration is an example of the Default Icon path/file name located in the Windows NT Registry:



Registry Editor - [HKEY_CLASSES_ROOT on Local Machine]

```
├ Excel.Sheet.5
│  ├ CLSID
│  ├ DefaultIcon
│  ├ DocObject
│  ├ Insertable
│  ├ protocol
│  └ shell
├ Excel.SLK
├ Excel.Template
```

<No Name> : REG_SZ : C:\MSOffice\Excel\excel.exe,1

See the "Windows User Interface Guidelines," chapter 10, for further details on registering file types and application data.

Required:
The application installer must register *all* shared components designed to be uninstalled under the following Registry key:

```
[HKEY_LOCAL_MACHINE]\SOFTWARE\Microsoft\Windows\Current
Version\SharedDLLs
```

Required:
Installers must fail gracefully in the case where a system component cannot be replaced because file system security prevents an existing file from being overwritten. Installers that need to overwrite system files, for example, should check up-front if the current user is a member of the administrators local group. This will avoid confusion later by preventing the user from getting file copy errors from which there is no recovery.

Required:
Applications must be tested on Windows NT under non-administrator accounts. An application must run under a user account but that user must not be able to change the application setup configuration, etc.

Required:
When installing the application, the installer must check for the user privilege level. This may be accomplished by using the Sample Code below or the KB article PSS ID# Q118626. If the user is not an administrator and this will cause significant differences in functionality, the installer must inform the user about the consequences of installing without admin privileges. If the application will work but with limited functionality, the installer must warn the user that only limited functionality will be available since they do not have admin privileges, and the installer must allow them to discontinue the installation.

Sample Code:

```
//------------------------------------------------------------//
//
// IsAdmin() - tests to see if the current user is an admin
//
//------------------------------------------------------------//

BOOL IsAdmin() {

  SC_HANDLE hSC;

  //
  // Try an Admin Privileged API - if it works return
  // TRUE - else FALSE
  //
        hSC = OpenSCManager(
                NULL,
                NULL,
                GENERIC_READ | GENERIC_WRITE | GENERIC_EXECUTE
                );

    if( hSC == NULL ) {
        return FALSE;
    }

  CloseServiceHandle( hSC );
  return TRUE;
}
```

Required:
The installer must *not* register components that are self registering, such as DirectX and Direct3D. The installers for these components will register their components as needed.

Required:
Core components must not be refcounted in the registry by application installers. Please see http://www.microsoft.com/windows/thirdparty/winlogo for the most up-to-date and complete list of Windows 95 and Windows NT core components.

Required:
The application must not add information to Win.ini or System.ini.

Exception:
Certain multimedia applications may require video codecs not supplied with Windows NT or 95. Products that must support links to existing 16-bit products may write to these files. Certain other products, such as screen savers, may not be supported in the Registry and must write to the .ini files. Writes to the embedded section of Win.ini may be unavoidable. It is okay to write to the fonts section of Win.ini and to leave behind fonts and font information after uninstallation.

Required:
All writes to Win.ini or System.ini must be explained in the Vendor Questionnaire.

Recommended:
If you don't think your application will run correctly without being installed with admin privileges, you should only give the user the choice to abort the installation.

Recommended:
If the user is allowed to complete the installation without admin privileges, the application, at start up time, should warn the user that only limited functionality will be available.

Recommended:
Applications are STRONGLY DISCOURAGED from overwriting core components. . Please see http://www.microsoft.com/windows/thirdparty/winlogo for the most up-to-date and complete list of Windows 95 and Windows NT core components. If this is absolutely necessary, the installer must not overwrite core components with versions that are older than those currently on the disk. The installer must use the dwFileVersion fields in the VS_FIXEDFILEINFO structure to make this determination.

Recommended:
The application should default installation to a directory under "drive:\Program Files". It is recommended that an application install *solely* under its "Program Files" directory. The application's installer should prevent an application from storing application DLLs or executables under the system root.

Recommended:
An application should be prepared to operate smoothly when the operating system is upgraded (for example, from Windows 95 to future versions of Windows NT). To do so, you should comply with the following:

- When installing on one 32-bit Microsoft Windows operating system, the installer should install all additional binaries that would be required to make the program operate on the other operating system, and determine which components to use at run time. For example, if the application requires a different DLL on Windows NT than on Windows 95, install both and pick the one to actually load and use at application startup time. If your application requires a different .exe for each operating system, then write a stub .exe that calls the correct .exe based on a run-time version check (GetVersionEx() Win32 API).
- Installers and applications should make no distinction between Windows NT and Windows 95 when writing to the registry. The structure of the registries on Win32 systems are similar enough to make this possible. For network applications, please refer to the MSDN information about the differences between the Windows 95 and Windows NT registries for network components.

- Installers and applications should store string data as separate REG_SZ, REG_MULTI_SZ or REG_EXPAND_SZ values instead of embedding such strings as REG_BINARY data in the registry. This helps resolve ANSI <-> Unicode conversion issues when upgrading a Windows 95-based machine to future versions of Windows NT.
- Installers and applications should make use of REG_EXPAND_SZ and the %SystemRoot%, %windir%, and %SystemDrive% environment variables. In particular, do NOT hardcode paths to the Windows system root or to the drive containing Windows in the registry since users may remap their drives under Windows NT. For example, under Windows NT:

```
MyPath : REG_SZ : "C:\Program Files\MyApp"       // bad

MyPath : REG_EXPAND_SZ :
"%SystemDrive%\ProgramFiles\MyApp" // good
```

- The Win32 API **ExpandEnvironmentStrings** can then be used when the string is retrieved to get the up-to-date path.
- If your program uses a file extension that is used by other applications (for example, .htm files), the installer program should check the registry and ask the user if they want to change the default program for that particular file extension.

4.4. Uninstall

Required:
✓ The product must provide a fully automated uninstaller that removes the program files, folders, and Registry entries for the 32-bit Microsoft Windows operating system.

Required:
✓ The uninstaller must be accessible through **Add/Remove Programs** on the Control Panel and must operate properly from the Control Panel.

Required:
✓ All files and folders copied onto the hard disk must be removed, with certain exceptions.

Exception:
User data files including the following should remain on the hard disk:

- Resources that other programs might use, such as sharable DLLs, sharable fonts, and sharable Registry entries.
- It is better to err on the side of *safety* regarding other applications. If you are not sure whether removing a DLL might harm other applications, it is better to leave it behind. However, you must explain everything you leave behind when you submit your application for testing. The proper place to do this is in the Vendor Questionnaire.

Tip: Remember to remove .gid files created by your Help. Create a zero-length (0) file with the same name at install time.

Required:
☑ The uninstaller must remove shortcuts placed anywhere in the **Start Menus** by its installer.

Required:
☑ The uninstaller must remove all Registry entries (with the exception of keys that might be shared by other programs). Uninstalling an icon registration is optional; removing Registry entries may leave leftover user data files without icons.

Required:
☑ The uninstaller must accurately decrement the count on all components your application uses that are installed as shared components.

Required:
☑ If the Reference Count is at zero (0) and you will not be removing the component, you must leave the Reference Count at 0. This requirement prevents installers from inaccurately incrementing the Shared Count if the component was on the system, but was not registered.

Recommended:
If your installer finds the component already on the system and not registered, the SharedDLL count should be incremented by 1 plus the number of clients being installed. For example, if you install your application with 3 clients using a shared component, your installer will bump the SharedDLL count by 3. But if the shared component was already on the system and no SharedDLL exists for it (that is, the previous installer did not create the refcount), then set the SharedDLL count to 4. That way, when your application is uninstalled, it leaves the shared component on the system with a refcount of 1.

Required:
☑ The uninstaller must *not* decrement or remove any core component, in particular Microsoft Foundation Class Library (MFC) DLLs, along with all other DLLs installed. Please see http://www.microsoft.com/windows/thirdparty/winlogo for the most up-to-date and complete listing of all Windows NT and Windows 95 core components.

Required:
☑ Due to prior installation issues, ODBC and DAO components (DLLs) must not be removed by your uninstaller.

Required:
☑ The uninstaller must remove itself.

Required:
☑ Certain uninstall executable files can and should be left behind if they may be shared by other programs.

Recommended:
Your uninstaller can choose to leave behind any shared component that you do not feel safe removing.

Recommended:
Uninstall programs should be 32-bit.

4.5. UI/Shell

Required:
☑ The application must use the system metrics for sizing wherever appropriate. Do not hard code pixel dimensions of menus, scroll bars, sizes of captions, border sizes, etc.

Verification:
A simple way to verify this is to click the **Appearance** tab on the Display box in Control Panel. Change some of the sizes to a large value (for example, change the font in the title bar or the size of scroll bars). You can select one of the extra large appearance schemes provided with the operating system, especially "Windows Standard (extra large)" and "High Contrast Black (extra large)". Then go back into your product and test the major screens, dialog boxes, and controls to ensure that these changes have been properly reflected, that is, they have not created anomalous display situations with wrongly hard-coded elements. For example, be sure that such things as the **OK** and **Cancel** buttons are still visible.

Tip: Use GetSystemMetrics to get the dimensions of menus, scroll bars, sizes of captions, border sizes, etc. To get the space left over after the tray takes up the screen, use:

GetSystemMetrics (SM_CXFULLSCREEN, SM_CYFULLSCREEN)

SystemParametersInfo (SPI_GETWORKAREA)

Recommended:
The main metrics your application should be aware of and their mappings with the system elements are as follows:

SM_CXBORDER	SM_CYBORDER	SM_CXMENUSIZE
SM_CYMENUSIZE	SM_CXSIZEFRAME	SM_CYSIZEFRAME
SM_CXSMICON	SM_CYSMICON	SM_CXSMSIZE
SM_CYSMSIZE	SM_CYSMCAPTION	SM_CXCURSOR
SM_CYCURSOR	SM_CXDLGFRAME	SM_CYDLGFRAME
SM_CXDOUBLECLK	SM_CYDOUBLECLK	SM_CXFRAME
SM_CYFRAME	SM_CXFULLSCREEN	SM_CYFULLSCREEN
SM_CXHSCROLL	SM_CYHSCROLL	SM_CXHTHUMB
SM_CXICON	SM_CYICON	SM_CXICONSPACING

SM_CYICONSPACING	SM_CXMIN	SM_CYMIN
SM_CXSCREEN	SM_CYSCREEN	SM_CXSIZE
SM_CYSIZE	SM_CXVSCROLL	SM_CYVSCROLL
SM_CYCAPTION	SM_CYKANJIWINDOW	SM_CYMENU
SM_CYVTHUMB	SM_CXDRAG	SM_CYDRAG
SM_CXEDGE	SM_CYEDGE	SM_CXFIXEDFRAME
SM_CYFIXEDFRAME	SM_CXMAXIMIZED	SM_CYMAXIMIZED
SM_CXMAXTRACK	SM_CYMAXTRACK	SM_CXMENUCHECK
SM_CYMENUCHECK	SM_CXMINIMIZED	SM_CYMINIMIZED
SM_CXMINSPACING	SM_CYMINSPACING	

Exception:
Certain windowable applications that use a bitmap for their window (and therefore have unique resizing considerations) may not be able to support the full range of font sizes and other pixel dimensions available to the user. Very large font sizes may, for example, push their title bars off the screen. These applications must still use system metrics, however, to the extent possible.

Exception:
Certain multimedia applications may have aesthetic and design considerations, such as custom dialog boxes and user interface (UI) elements. These will be judged on a case-by-case basis. If you are unsure of whether or not this requirement applies, contact the testing organization or Microsoft. You must detail these exceptions in your Vendor Questionnaire.

Recommended:
Full-screen (that is, not windowable) multimedia and game applications should use system metrics.

Recommended:
Avoid hard coding any fonts sizes smaller than 10 points. Small fonts can cause eye strain and may be difficult for many people to read.

Recommended:
The application should be compatible with changes to the system font size and changes to the number of pixels per logical inch. For additional information, see "Size" in paragraph 7.3.3.

Recommended:
Applications should use the system colors wherever appropriate and allow the user to customize all other colors. System colors should always be used in their proper foreground/background combinations. For more information, see "Color" in paragraph 7.3.4.

Recommended:
Applications should support the High Contrast option, which indicates that the user wants a high degree of legibility. For more information, see "Color" in paragraph 7.3.4.

Exception:
Exceptions are made for games and inherently graphical applications such as image editing tools. All exceptions must be explained in the Vendor Questionnaire.

Recommended:
The application should not convey important information by color alone. If color alone is the default method for conveying the information, the application should provide an option to convey this information by other means.

Recommended:
The application should provide keyboard access to all features. For more information, see "Keyboard User Interface" in paragraph 7.3.5.

Exception:
Exceptions may be made in the following cases:

- Games that are dependent upon the use of a mouse or joystick. It is recommended, but not required, that games provide keyboard support where feasible. They should not require a pointing device except where it is unavoidable due to the nature of the game.
- CAD and similar applications that are dependent upon graphing tablets and other input devices.
- Applications that are entirely graphical, such as image editors, may rely on the MouseKeys() feature built into the 32-bit Windows operating systems to allow users to move the mouse pointer. However, this is not acceptable for drawing applications that allow the user to independently manipulate separate text and graphic objects.

Exceptions may also be made for minor features that are not required for operation of the program, when the major features in the application have keyboard access. These situations will be judged on a case-by-case basis. All major features or functional areas without keyboard access should be explained in the Vendor Questionnaire.

Recommended:
The keyboard user interface should be fully documented.

Exception:
It is not necessary to document elements that simply follow the Windows conventions for elements such as standard menus and controls.

Tip: This information may be provided in your standard documentation that is included with the product, or it may be orderable separately. In the latter case, it is strongly recommended that the standard documentation inform the user that this additional documentation is available.

Recommended:
Notify other software of the locations of the keyboard focus and selection. For more information, see "Exposing Focus and Selection" in paragraph 7.3.6.

Recommended:
The application should allow other software to identify and manipulate all screen elements that the user interacts with. For more information, see "Exposing Screen Elements" in paragraph 7.3.7.

Recommended:
Allow the user to customize all user interface timings that are not based on standard system metrics. For more information, see "Timings" in paragraph 7.3.8.

Exception:
Exceptions may be made for minor features that are not required for operation of the program. This situation will be judged on a case-by-case basis. All exceptions must be explained in the Vendor Questionnaire.

Recommended:
Don't trigger unexpected side effects based on changes in pointer or keyboard focus locations. For more information, see "Unexpected Side Effects" in paragraph 7.3.9.

Recommended:
Do not convey any important information by sound alone. If sound alone is the default method for conveying the information, the application should provide an option to convey this information by other means. For more information, see "Sound" in paragraph 7.3.10.

Recommended:
You should allow the user to choose font names and font sizes wherever it is appropriate.

4.6. Support for the Internet

Required:
✓ Authors of ActiveX™ Controls must digitally sign their controls, as specified in the ActiveX SDK. Controls should also be marked "safe for persistence" or "safe for scripting", as appropriate. For more information, see the ActiveX SDK at http://www.microsoft.com/IntDev/sdk.

Recommended:
Authors of ActiveX Controls should provide support for automatic code download. If the control is a single file, then no additional work is required. If multiple files are required, then a .cab, .inf, or a self-extracting executable should be provided. Refer to the Internet Component Download spec in the ActiveX SDK.

Recommended:
Applications, tools, controls, or any other software that installs or sets up DLLs should honor the ModuleUsage registry section described in the appendix of the ActiveX SDK.

Recommended:
Your application should expose the Help files as HTML. Over time, Microsoft will phase out support for the WinHelp engine, and replace it with Web-based help.

Recommended:
Your application should have an FTP save option, enabling a user to save to an FTP site in the same way as they can save to the local drive or to a network drive.

Recommended:
Your application should produce documents that can be viewed from a browser. You can do this either by implementing a "Save As HTML" feature, or by adding ActiveX Document (docobj) support to your application. See the ActiveX SDK for more information about adding ActiveX Documents.

4.7. Stability and Functionality

Required:
☑ The product must be fully functional and stable on the most recent versions of both the Windows NT and Windows 95 operating systems.

Required:
☑ The product must be stable, and all functionality must be in place when it is submitted for testing. While we are not performing QA testing on your product for Logo purposes, we will do some basic stability testing to confirm that your product does not appear to adversely affect the overall stability of Windows NT or Windows 95.

Required:
☑ If your product includes online help, it must be fully functional at the time of testing. However, content does not matter.

Required:
☑ The application must support either Alt-Tab or Ctrl-Esc to switch focus from full screen to the desktop.

Verification:
The following are some of the tests we will run. You can run them yourself to pretest your product:

ON WINDOWS 95

1. Find the Stress application in the Win32 SDK (\mstools\bin\win95\stress.exe), and select the following options: